Google AI’s OR-Tools CP-SAT Solver

unofficial tutorial

Bobak Pezeshki
General Info

Home Page:  https://developers.google.com/optimization

About:  https://developers.google.com/optimization/introduction/overview
CP-SAT Solver

Overview Page:  https://developers.google.com/optimization/cp

Intro Use Case:  https://developers.google.com/optimization/cp/cp_solver

Documentation:
https://developers.google.com/optimization/reference/python/sat/python/cp_model
Simple Example
Problem Description

Consider three numbers.

Each can be either 0, 1, or 2.

The first two numbers are the same.

What are the different possibilities for the three numbers?
Constraint Model

Variables: \( V = \{ a, b, c \} \)

Domains: \( D = \{ D_a, D_b, D_c \} \), \( \text{st. } \forall x \in V, \ D_x = \{ 0, 1, 2 \} \)

Constraints: \( a == b \)

Primal Graph:

```
  a -- b -- c
```
from ortools.sat.python import cp_model

text = "Hello World"

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a, b, c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('All solutions found!')
elif status == cp_model.FEASIBLE:
    print('Some solutions found!')
else:
    print('No solution could be found!')


"Hello World"
Import the CP-SAT Package

```python
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a, b, c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('
' + "All solutions found!" + '
')
elif status == cp_model.FEASIBLE:
    print('
' + "Some solutions found!" + '
')
else:
    print('
' + "No solution could be found!" + '
')
```
Create Model Object

```python
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a, b, c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('All solutions found!

elif status == cp_model.FEASIBLE:
    print('Some solutions found!

else:
    print('No solution could be found!

```
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a, b, c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('
' + "All solutions found!
"
)
elif status == cp_model.FEASIBLE:
    print('
' + "Some solutions found!
"
)
else:
    print('
' + "No solution could be found!
"
Add Constraints to Model

```python
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a,b,c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('
' + "All solutions found!" + '
')
elif status == cp_model.FEASIBLE:
    print('
' + "Some solutions found!" + '
')
else:
    print('
' + "No solution could be found!" + '
')
```
Create CP-SAT Solver Object

```python
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a, b, c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('\n' + "All solutions found!" + '\n')
elif status == cp_model.FEASIBLE:
    print('\n' + "Some solutions found!" + '\n')
else:
    print('\n' + "No solution could be found!" + '\n')
```
Create A Solution Printer Object (optional)

```python
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a,b,c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('
' + 'All solutions found!' + '
')
elif status == cp_model.FEASIBLE:
    print('
' + 'Some solutions found!' + '
')
else:
    print('
' + 'No solution could be found!' + '
')
```
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a,b,c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('All solutions found!'
else:
    print('No solution could be found!' +

from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a, b, c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('
' + "All solutions found!" + '
')
elif status == cp_model.FEASIBLE:
    print('
' + "Some solutions found!" + '
')
else:
    print('
' + "No solution could be found!" + '
')
from ortools.sat.python import cp_model

model = cp_model.CpModel()

a = model.NewBoolVar('a')
b = model.NewBoolVar('b')
c = model.NewBoolVar('c')

model.Add(a == b)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a,b,c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('
' + "All solutions found!" + '
')
elif status == cp_model.FEASIBLE:
    print('
' + "Some solutions found!" + '
')
else:
    print('
' + "No solution could be found!" + '
')
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.AddAllDifferent([a, b, c])

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a, b, c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('\n' + "All solutions found!" + '\n')
elif status == cp_model.FEASIBLE:
    print('\n' + "Some solutions found!" + '\n')
else:
    print('\n' + "No solution could be found!" + '\n')
Element Constraints

from ortools.sat.python import cp_model

cp_model = cp_model.CpModel()

num_vals = 3
a = cp_model.NewIntVar(0, num_vals - 1, 'a')
b = cp_model.NewIntVar(0, num_vals - 1, 'b')
c = cp_model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

E_ac = [2, 1, 0]
model.AddElement(a, E_ac, c)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a, b, c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('All solutions found!'
elif status == cp_model.FEASIBLE:
    print('Some solutions found!'
else:
    print('No solution could be found!')
Relational Constraints

```python
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)

R_ac = [(0,2),(1,1),(2,1),(2,2)]
model.AddAllowedAssignments([a,c],R_ac)

solver = cp_model.CpSolver()
printer = cp_model.VarArraySolutionPrinter([a,b,c])
status = solver.SearchForAllSolutions(model, printer)

if status == cp_model.OPTIMAL:
    print('All solutions found!')
elif status == cp_model.FEASIBLE:
    print('Some solutions found!')
else:
    print('No solution could be found!')
```

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
from ortools.sat.python import cp_model

model = cp_model.CpModel()

num_vals = 3
a = model.NewIntVar(0, num_vals - 1, 'a')
b = model.NewIntVar(0, num_vals - 1, 'b')
c = model.NewIntVar(0, num_vals - 1, 'c')

model.Add(a == b)
model.Maximize(2*c - a)
solver = cp_model.CpSolver()
printer = cp_model.VarArrayAndObjectiveSolutionPrinter([a, b, c])
status = solver.SolveWithSolutionCallback(model, printer)

if status == cp_model.OPTIMAL:
    print('
' + "Optimal solution found!" + '
')
elif status == cp_model.FEASIBLE:
    print('
' + "A solution found, but may not be optimal." + '
')
else:
    print('
' + "No solution found!" + '
')
Extra Practice
Practice Satisfiability Problem

There are three instructors.

Each be teaching one class.

The university only has one room left available with four different time slots.

Model this as a constraint programing problem and print all solutions.
Practice Optimization Problem

Consider the instructor assignment problem from before...

For each professor, the university has also recorded the various time-slot preferences, each of which they assign a preference multiplier to.

<table>
<thead>
<tr>
<th></th>
<th>Time-Slot 1</th>
<th>Time-Slot 2</th>
<th>Time-Slot 3</th>
<th>Time-Slot 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instr 1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Instr 2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Instr 3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Practice Optimization Problem

Consider the instructor assignment problem from before...

For each professor, the university has also recorded the various time-slot preferences, each of which they assign a preference multiplier to.

Additionally, different professors have different seniority, again corresponding to different multipliers.

<table>
<thead>
<tr>
<th></th>
<th>Seniority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instr 1</td>
<td>5</td>
</tr>
<tr>
<td>Instr 2</td>
<td>4</td>
</tr>
<tr>
<td>Instr 3</td>
<td>3</td>
</tr>
</tbody>
</table>
Practice Optimization Problem

Consider the instructor assignment problem from before...

For each professor, the university has also recorded the various time-slot preferences, each of which they assign a preference multiplier to.

Additionally, different professors have different seniority, again corresponding to different multipliers.

Using the different multipliers, model this as a constraint programming problem and print the solution that optimizes the sum of product of each professor’s time-slot preference multiplier and seniority multiplier.
Other Notes
Many more kinds of constraints and functions

https://developers.google.com/optimization/reference/python/sat/python/cp_model
#top_of_page
You can limit the solver’s time or number of solutions

https://developers.google.com/optimization/cp/cp_tasks
Default Solution Printers use Zero-Based Indexing

Solution 0, time = 0.02 s
  a = 2   b = 2   c = 0
Solution 1, time = 0.02 s
  a = 1   b = 1   c = 0
Solution 2, time = 0.05 s
  a = 1   b = 1   c = 1
Solution 3, time = 0.05 s
  a = 2   b = 2   c = 1
Solution 4, time = 0.06 s
  a = 2   b = 2   c = 2
Solution 5, time = 0.09 s
  a = 1   b = 1   c = 2
Solution 6, time = 0.09 s
  a = 0   b = 0   c = 2
Solution 7, time = 0.10 s
  a = 0   b = 0   c = 1
Solution 8, time = 0.10 s
  a = 0   b = 0   c = 0

Note there are actually 9 solutions!
You Can Create Your Own Solution Collector Class

Need to be derived from the CpSolverSolutionCallback class.

Ex: (from: https://stackoverflow.com/questions/58934609/obtain-list-of-sat-solutions-from-ortools)

class VarArraySolutionCollector(cp_model.CpSolverSolutionCallback):

    def __init__(self, variables):
        cp_model.CpSolverSolutionCallback.__init__(self)
        self.__variables = variables
        self.solution_list = []

    def on_solution_callback(self):
        self.solution_list.append([self.Value(v) for v in self.__variables])